**Ledesma Ayala Angel Yeremi**

# Servlet y JSP:

**Servlet**: Un Servlet es un componente de programación en Java que se utiliza para procesar solicitudes y respuestas en el lado del servidor en aplicaciones web. Funciona como un controlador que gestiona la lógica empresarial y la interacción con el cliente en un entorno web. Los Servlets son compatibles con la plataforma Java EE (Enterprise Edition) y se ejecutan en un servidor web o de aplicaciones. Su función principal es procesar solicitudes HTTP, lo que les permite realizar tareas como recopilar datos de formularios, interactuar con bases de datos y generar respuestas dinámicas para los usuarios.

**JSP (JavaServer Pages):** JSP es una tecnología de Java que se utiliza para crear páginas web dinámicas. A diferencia de los Servlets, las JSP se centran principalmente en la presentación y la interfaz de usuario. Las páginas JSP son documentos HTML que pueden contener fragmentos de código Java incrustados. Esto permite que los desarrolladores mezclen contenido estático (HTML, XML) con contenido dinámico generado por código Java. Las JSP son especialmente útiles para separar la lógica de presentación del código de lógica empresarial en aplicaciones web, lo que facilita el mantenimiento y la escalabilidad.

**Creación y Configuración de Servlets:**

La creación y configuración de Servlets es una parte fundamental en el desarrollo de aplicaciones web en Java. Los Servlets son componentes que se utilizan para procesar solicitudes y respuestas en el lado del servidor. Aquí se explica cómo crear y configurar Servlets:

1 **Creación y Configuración de un Servlet en Jakarta EE**

Para crear un servlet en Jakarta EE, sigue estos pasos:

- Extender la Clase HttpServlet: Crea una clase Java que extienda jakarta.servlet.http.HttpServlet.
- Implementar Métodos para Manejar Solicitudes HTTP: Anula los métodos como doGet() o doPost(), según el tipo de solicitud que tu servlet manejará. Por ejemplo, para manejar solicitudes GET, sobrescribe doGet().

```
import jakarta.servlet.*;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.*;

import java.io.IOException;

import java.io.PrintWriter;

@WebServlet("/hello") public class HelloWorldServlet extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.println("Hello, World!");
    }
}
```

- Anotación @WebServlet: Utiliza esta anotación para definir la configuración y el mapeo URL de tu clase Servlet. Por ejemplo, @WebServlet("/hello") indica que este servlet responderá a solicitudes HTTP que apunten al camino /hello.

5. Despliegue del Servlet

- Empaquetado y Despliegue: Una vez que hayas creado tu servlet, empaquétalo junto con tu aplicación web en un archivo WAR (Web Application Archive).
- Despliegue en Servidor: Despliega el archivo WAR en un servidor que soporte Jakarta EE, como WildFly o Tomcat.

**Desarrollo de Páginas Web con JSP:**

El desarrollo de páginas web con JavaServer Pages (JSP) es una parte esencial en la creación de aplicaciones web dinámicas utilizando tecnologías Java. JSP permite combinar código Java con contenido HTML para generar páginas web interactivas y personalizadas. Aquí se explica cómo desarrollar páginas web con JSP:

1. Creación de una Página JSP:

- Para comenzar, crea un archivo con extensión ".jsp", por ejemplo, "miPagina.jsp". Este archivo contendrá tanto contenido HTML estático como fragmentos de código Java incrustados.

2. Incorporación de Código Java:

- Puedes insertar código Java en tu página JSP utilizando etiquetas <% %> o <%= %>.
- También puedes utilizar declaraciones <%! %> para declarar variables y métodos que estarán disponibles en todo el alcance de la página JSP.

3. Uso de Objetos Implícitos:

- JSP proporciona objetos implícitos que simplifican el acceso a información web común, como solicitudes, sesiones y contexto de aplicación. Algunos de los objetos implícitos más comunes son:
    - request: Representa la solicitud HTTP actual.
    - session: Representa la sesión del usuario.
    - application: Representa el contexto de la aplicación.
    - out: Permite imprimir contenido en la respuesta HTTP.
- Por ejemplo, puedes acceder a parámetros de solicitud y mostrarlos en tu página JSP.

4. Estructura y Diseño HTML:

- Combina el contenido HTML estático con el código Java para crear la estructura y el diseño de tu página web. Puedes incluir etiquetas HTML, CSS y JavaScript según tus necesidades.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Mi Página JSP</title>
    </head>
    <body>
        <h1>Bienvenido a mi página JSP</h1>
        <% // Código Java para la lógica dinámica String mensaje = "¡Hola, mundo!"; out.println(mensaje); %>
    </body>
</html>
```

5. Interacción con Bases de Datos:

- Puedes utilizar JSP para interactuar con bases de datos y mostrar datos dinámicos en tu página web. Esto suele involucrar el uso de Java Database Connectivity (JDBC) u otros marcos de trabajo de persistencia de datos.

6. Validación de Formularios y Control de Flujo:

- JSP te permite validar formularios y controlar el flujo de tu aplicación web utilizando estructuras de control Java, como condicionales (if-else) y bucles (for, while).

7. Despliegue de la Aplicación:

- Empaqueta tu aplicación web que contiene las páginas JSP en un archivo WAR y despliégala en un servidor web o de aplicaciones Java.

**Comunicación entre Servlets y JSP:**

La comunicación entre Servlets y JSP es una parte fundamental en el desarrollo de aplicaciones web dinámicas en Java. Servlets gestionan la lógica empresarial y procesan las solicitudes del cliente, mientras que JSP se utilizan para la presentación y la generación de contenido HTML. Aquí se explica cómo ocurre esta comunicación:

1. Solicitud Inicial al Servlet:

- El proceso comienza cuando un cliente (navegador web) realiza una solicitud HTTP a una URL específica que está mapeada a un Servlet en el archivo "web.xml" de la aplicación web.

2. Procesamiento del Servlet:

- El contenedor de Servlets (como Tomcat) recibe la solicitud y la asigna al Servlet correspondiente según la configuración en "web.xml". El Servlet procesa la solicitud, realiza la lógica empresarial necesaria y puede preparar datos que se mostrarán en la página web.

3. Preparación de Datos para JSP:

- El Servlet puede almacenar datos en el ámbito de solicitud, sesión o aplicación, o puede adjuntar datos directamente a la solicitud HTTP como atributos. Estos datos son los que se utilizarán en la página JSP para generar contenido dinámico.

4. Redirección a la Página JSP:

- Una vez que el Servlet ha procesado la solicitud y preparado los datos, puede redirigir al cliente a una página JSP para la presentación de resultados. Esto se hace configurando una respuesta HTTP de redirección o mediante una redirección interna en el código del Servlet.

5. Procesamiento de la Página JSP:

- La página JSP recibe la solicitud y tiene acceso a los datos preparados por el Servlet. Utilizando etiquetas JSP y fragmentos de código Java, la página JSP puede acceder a estos datos y generar contenido HTML dinámico que se enviará al cliente como respuesta.

6. Respuesta al Cliente:

- La página JSP genera la respuesta HTML completa que se envía al cliente, y el cliente la muestra en su navegador web.

7. Interacción Continua:

- La interacción entre Servlets y JSP puede ser continua a medida que el usuario interactúa con la aplicación web. Por ejemplo, el usuario podría enviar un formulario a un Servlet, que procesa los datos y redirige a una página JSP para mostrar una confirmación.

8. Comunicación Bidireccional:

- La comunicación entre Servlets y JSP es bidireccional. Los Servlets pueden pasar datos y atributos a las JSP para su presentación, y las JSP pueden enviar datos de vuelta a los Servlets, por ejemplo, al enviar un formulario.

En resumen, Servlets y JSP trabajan juntos en aplicaciones web Java para separar la lógica empresarial de la presentación. Los Servlets gestionan la lógica y preparan datos, mientras que las JSP se encargan de la presentación y generan contenido HTML dinámico basado en los datos proporcionados por los Servlets. Esta comunicación colaborativa permite crear aplicaciones web interactivas y personalizadas.